Problem: The creation of what one would consider to be “fair” teams in a class is a challenge. One instructor believes that numbers are the least biased method of creating “fair” teams and will use exam scores to assign students to their teams. All teams include two students and given an even number of exam scores (up to 50) display the maximum difference in terms of points from the strongest (in terms of total exam score of the partners) and the weakest teams based on their total exam score.

NOTE: THE USE OF USER-DEFINED FUNCTIONS IS REQUIRED ON THIS ASSIGNMENT.

Example Execution #1:
Enter exam scores now: 85 60 100 65 -1

Max Difference: 10

• **Explanation:** There are only four students in this class. The first team has a total of 160 points and the second a total of 150. A -1 value is entered to terminate input when the number of exams entered is less than 50.

Example Execution #2:
Enter exam scores now: 60 80 55 90 65 88 -1

Max Difference: 3

• **Explanation:** There are three teams in this example with totals of 148, 145, and 145.

Example Execution #3:
Enter exam scores now: 45 60 75 77 65 90 70 88 92 68 48 51 63 93 93 66 74 90 87 57 45 60 75 77 65 90 70 88 92 68 48 51 63 93 93 66 74 90 87 57

Max Difference: 15

• **Explanation:** When 50 scores are entered there is no -1 value needed to terminate input.

Example Execution #4:
Enter exam scores now: 77 87 91 43 55 -1

Error! Number of scores must be even!!
Enter exam scores now: 77 87 91 43 55 27 45 -1

Error! Number of scores must be even!!
Enter exam scores now: 77 87 91 43 55 27 45 55 87 69 -1

Max Difference: 14

**Academic Integrity Reminder:**

• Please review the policies of the course as they relate to academic integrity. The assignment you submit should be your own original work. You should be consulting only course staff regarding your specific algorithm for assistance. Collaboration is not permitted on individual homework assignments.
Additional Requirements:

- Accept input and produce output exactly as seen in the example executions. Each example provided on the previous page represents a single execution of your program.
  - The final example demonstrates the input validation requirements.
    - The number of exam scores must be even.
    - All exam scores and their sums will fit within the int data type.
  - Do not add any “bonus” features not demonstrated in the example executions provided.
- Your program MUST make good use of user-defined functions. It is up to you to determine how to factor the tasks in this program into individual functions. Revisit the course standards and expectations of the main function now that user-defined functions are a requirement of this and all future assignments.
  - DO NOT use any material found outside of the first EIGHT chapters of the C text.
    - You may adapt the sorting algorithms as they appear in the book or notes as long as you reference your source in the assignment function header and bring all code in compliance with course programming and documentation standards.
- A program MUST compile to be considered for partial credit. The submission script will reject the submission of any file that does not compile.

Course Programming and Documentation Standards Reminders:

- The penalty for violating course standards (commenting, indenting, use of { and } spaces between operators and operands, use of symbolic/defined constants) will increase as the semester continues.
- Include the appropriate assignment header (head_hw) and complete the header prior to making your final submission. The header should not inhibit the ability to compile your program.
- Use the course function header (head_fx) for every user-defined function in your program.
  - List and comment all parameters to a function, one per line, in the course function header.
  - All function declarations will appear in the global declaration section of your program.
  - The user-defined function definitions will appear in your program after main.
- Place a single space between all operators and operands.
- Comment all variables to the right of each declaration. Declare only one variable per line.
- Select meaningful identifiers (names) for all variables and functions in your program.
- Use { and } with all selection and repetition constructs in your program.
- Indent all code found within the body of main and all functions exactly two spaces.
  - You must also do this for selection and repetition constructs that have a body.
- Do not single (or double) space the entire program, use blank lines when appropriate.

When you submit... only the last attempt of a submission is kept for grading. All other submissions are over-written and cannot be recovered. You may make multiple submissions but only the last attempt is retained and graded.

- Verify in the e-mail sent to you by the course that you have submitted the correct file, to the correct assignment (hw06), and to the correct lab section. Forwarding course e-mails from Purdue to external e-mail services may result in the mail being undelivered or end up being identified as spam.
- Leave time prior to the due date to seek assistance should you experience difficulties completing or submitting this assignment.
- All attempts to submit via a method other than through the sage server as set up during the first week of the semester will be denied consideration.

Assignment deadlines... are firm and the electronic submission will disable promptly as advertised. We can only grade what you submit as expected (including the correct assignment and lab section) prior to the assignment deadline.

All course programming and documentation standards are in effect for this and each assignment this semester. Please review this document in your course notes packet.